Oracle9i 中 JAVA 和 PL/SQL 的互操作方法

A Interoperation Method Between JAVA and PL/SQL In Oracle 9i

姚世军（郑州市解放军信息工程大学理学院 450052）

摘要：存储过程是 Oracle9i 应用程序中广泛使用的服务器端程序。Oracle9i 中可以使用 PL/SQL 语言和 JAVA 语言来建立存储过程。本文介绍了这两类存储过程的建立方法，特别用详细的示例介绍了它们之间的互操作方法。

关键词：存储过程　JAVA 类　PL/SQL 块

1　引言

Oracle9i 在 Java 和 PL/SQL 中提供了方便的互操作性。PL/SQL 程序块可以透明地调用 JAVA 存储过程，可以建立在 PL/SQL 中调用的 EJB 或 CORBA 组件。通过包裹程序(WRAPPER)将使 PL/SQL 程序透明地访问 JAVA 类库。Java 程序可以通过 JDBC 或 SQLJ 方便地调用 PL/SQL 存储过程，这样可以在与数据库敏感的地方使用 PL/SQL 存储过程。PL/SQL 和 JVM 紧密地集成在一起共享数据引擎的公共服务，从而使结构化性能大大改善。

总之，PL/SQL 语言和 JAVA 语言在数据库内可以无缝互操作，JAVA 扩展 PL/SQL 应用程序逻辑，JAVA 存储过程可以在服务器软件内调用 PL/SQL 存储过程。

2　开发 JAVA 存储过程的步骤

PL/SQL 存储过程只需用 CREATE PROCEDURE 或者 CREATE FUNCITON 等语句按照 PL/SQL 语言编程即可。JAVA 存储过程是发布给 SQL 并按模式对象存储在 ORACLE 数据库中 JAVA 类的方法，它的建立相对复杂，下面将简单介绍建立 JAVA 存储过程步骤。

2.1　建立或重用 JAVA 类

可以使用任何 JAVA 开发工具，建立满足用户需求的类，或者是重用现有的类。对类进行编译调试保证没有任何错误后，可以将类编译成以 CLASS 为扩展名的字节代码。

2.2　装载和解析 JAVA 类

使用 Oracle9i 中的命令行工具 loadjava 将 JAVA 源程序、JAVA 类和资源文件装入到 ORACLE 数据库中，它们都是以模式对象形式存储。如：

```
E:》loadjava -user scott/tiger Oscar.class;
```

上面命令将 OSCAR 类装载到 SCOTT 用户模式中。当调用 OSCAR 类中的方法时，服务器将用解析程序查询支持的函数类。

通过查询 USER_OBJECTS 数据字典视图可以确定指定的类是否装入到数据库中。

2.3　发布 JAVA 类

对于每个被 SQL 语句调用的 JAVA 方法，必须在数据字典中定义一个调用说明，即发布类。在装载类时，类的方法并不自动发布。对于给定的 JAVA 方法，如果该方法有返回值，那么要用 CREATE FUNCTION 或 CREATE PROCEDURE 语句定义一个函数或过程，即将类的方法发布为函数或过程；如果方法是 VOID 类型，那么只能将方法发布为过程。在函数或过程体内必须包括 LANGUAGE JAVA 语句。

下面例子发布 OSCAR 类的 QUOTE 方法，返回类型为 VARCHAR2。

```
SQL> CREATE FUNCTION oscar_quote RETURN VARCHAR2
2 AS LANGUAGE JAVA
3 NAME 'Oscar.quote() return java.lang.String';
```

2.4　调用 JAVA 存储过程

可以从 SQL 语言的 DML 语句、PL/SQL 块或 PL/SQL 子程序中调用已发布的 JAVA 存储过程。使用 SQL 语言的 CALL 语句可以在 SQL Plus 中或数据库触发器中调用 JAVA 存储过程。调用方法与其它 PL/SQL 存储过程一样。

```
CALL 过程名(参数1,参数2,……);
SQL>CALL updateSalary(1404, 50000);
```

3　从 JAVA 程序中调用 PL/SQL 存储过程

使用 JDBC 或 SQLJ 可以从 JAVA 程序中调用 PL SQL 存储过程。下例是先用 PL/SQL 建立存储函数 BALANCE，然后在 JAVA 程序中调用它：

```
FUNCTION balance (acct_id NUMBER) RETURN NUMBER IS
  acct_bal NUMBER; --用 PL/SQL 定义存储函数
  BEGIN
   SELECT bal INTO acct_bal FROM accts
   WHERE acct_no = acct_id; --查询指定帐号的值
   RETURN acct_bal; --函数返回值
```

```
END;
```

使用 JDBC 中调用上面的 PL/SQL 函数的语法：

```
CallableStatement cstmt =
   conn. prepareCall ( "{? = CALL balance(?)}"); //准备可调用语句
cstmt. registerOutParameter (1, Types.FLOAT); //注册 OUT 参数类型
cstmt.setInt(2, acctNo); //指定输入参数的值
cstmt.executeUpdate(); //执行
float acctBal = cstmt.getFloat(1); //返回执行结果
```

在 SQLJ 程序中调用上面 PL/SQL 存储过程的方法如下：

```
#sql acctBal = {VALUES(balance(:IN acctNo))};
```

4 从 PL/SQL 存储过程中调用 JAVA 存储过程

可以从任何 PL/SQL 块、子程序或包中调用 JAVA 存储过程。JAVA 类以模式对象存储在 ORACLE 数据库中，有权限的用户可以随时调用执行。

下面例子建立一个 JAVA 类：

```
import java.sql.*;
import oracle.jdbc.*;
public class Adjuster {
  public static void raiseSalary (int empNo, float percent) //定义方法
   throws SQLException {
   Connection conn =
   DriverManager.getConnection("jdbc:default:connection:"); //建立连接
   String sql = "UPDATE emp SET sal = sal * ? WHERE empno = ?"; //定义语句
   try {
     PreparedStatement pstmt = conn.prepareStatement(sql); //准备语句
     pstmt.setFloat(1, (1 + percent /100)); //参数赋值
     pstmt.setInt(2, empNo);
     pstmt.executeUpdate(); //执行语句
     pstmt.close(); //关闭
   } catch (SQLException e ) //异常处理
  {System.err.println(e.getMessage());}
}
```

类 Adjuster 的方法 raiseSalary 按指定的百分比提高指定雇员的工资。按照本文(2)中介绍的步骤，要在 PL/SQL 中使用 JAVA 存储过程，必须先发布它。由于 raiseSalary 是 VOID 类型的方法，所以将其发布为过程：

```
CREATE OR REPLACE PROCEDURE raise_salary (empno NUMBER, pct NUMBER)
  AS LANGUAGE JAVA
  NAME 'Adjuster.raiseSalary(int, float)';
```

从 PL/SQL 的无名块中调用过程 raise_salary：

```
DECLARE
  emp_id NUMBER;
  percent NUMBER;
BEGIN
  raise_salary(emp_id, percent); --调用已发布的 JAVA 存储过程
  ...
END;
```

下面的 JAVA 类 rowCount 返回给定数据库表的行数：

```
import java.sql.*;
import java.io.*;
import oracle.jdbc.*;
public class RowCounter {
public static int rowCount (String tabName) //定义方法
  throws SQLException {
  Connection conn = DriverManager.getConnection("jdbc:default:connection:");
   String sql = "SELECT COUNT(*) FROM " + tabName;
   int rows = 0;
   try {
     Statement stmt = conn.createStatement(); //建立语句
     ResultSet rset = stmt.executeQuery(sql); //执行查询
     while (rset.next()) {rows = rset.getInt(1);} //统计行数
     rset.close();
     stmt.close();
   } catch (SQLException e) //异常处理
    {System.err.println(e.getMessage());}
  return rows;
```

（下转第 70 页）

(上接第 67 页)

```
    }
}
```

发布类 RowCounter 中的 RowCount 方法，此时调用说明中定义返回类型为 NUMBER，不能用 INTEGER，因为调用说明中不能使用 NUMBER 的子类型 INTEGER、REAL 或 POSITIVE 等。

```
CREATE FUNCTION row_count (tab_name VARCHAR2) RETURN NUMBER
AS LANGUAGE JAVA
NAME 'RowCounter.rowCount(java.lang.String) return int';
```

可以从独立 PL/SQL 存储过程中调用函数 ROW_COUNT：

```
CREATE PROCEDURE calc_bonus (emp_id NUMBER, bonus OUT NUMBER)
AS emp_count NUMBER;

BEGIN
    emp_count := row_count('emp'); --调用发布的 JAVA 存储过程
    ...
END;
```

5 结束语

在 Oracle9i 应用程序开发中，存储过程能提供更好的性能、更高的效率和更大的可移植性。合理地将 JAVA 存储过程和 PL/SQL 存储过程有机的结合起来，充分发挥各自的优点，将大大提高应用程序的效率和质量，同时也将增加程序的可重用性。

参考文献

1 ORACEL 公司，Java Stored Procedures Developer's Guide。

2 ORACLE 公司，SQLJ Developer's Guide and Reference。

3 ORACLE 公司，JDBC Developer's Guide and Reference。